Using Git to submit assignments

Initial checkout

First, you **must** check out your directories from the repository. This will give you your own working copy of the files. On the GL systems, you'd use a command like this

```
git clone /afs/umbc.edu/users/s/i/simsek/pub/class_name/username.git directory_name_of_your_choice
```

For example, if you were in 411 and your username was xyzzy1, you might do this:

```
git clone /afs/umbc.edu/users/s/i/simsek/pub/411/xyzzy1.git cmsc411
```

This will create a directory called `cmsc411` containing a new working copy of your files. This is the only time you need to tell it where the repository is. After the intial checkout, once you `cd` into your working directory, Git can figure out which repository to use. All of the major Git commands apply to the current directory and all directories under it.

For remote access, you'd use something like this for your Git clone instead:

```
git clone username@gl.umbc.edu:/afs/umbc.edu/users/s/i/simsek/pub/class_name/username.git directory_name_of_your_choice
```

Editing

Edit away... No need to do anything special

Adding, removing, and ignoring files

Ask Git about the files it doesn't know about yet

```
git status
```

You may see sections for files "Changed but not updated" (including files previously checked in that have been modified or deleted), and/or for "Untracked files". If there are any files listed that should **not** be checked in (executables, object files, etc.), edit the `.gitignore` file and add them to the list of files Git should not track. This includes any files generated when you build — if you didn't create it, you probably don't want to check it in. Once `git status` just lists the files that should be checked in, you have tell Git that you really do mean to remove any that are marked to be deleted:

```
git rm file
```

Now tell Git what other files you want to commit (for new or changed files)

```
git add file
```

This is called "staging" your commit. Now you can commit a local copy of your work. This is **not yet submitted**, but you can use these intermediate revisions to track your work and go back to prior versions. You should give a short but useful message that will identify the main content of this commit. For example, "Initial parsing code", or "Fixed cross-product bug", etc.

```
git commit -m "short message"
```

Getting updates

If I make any changes to the initial sample files (or add some for the later assignments), you can get these using

```
git pull
```

Previous revisions

To see the set of commits, use `git log`

```
git log    # all commits
git log file   # commits for one file
```

Each log incudes a long hexadecimal number, which is the commit ID. To identify a commit, you only need to use enough of this number to be unique, usually the first 6-8 digits is enough.

To compare your current version of a file with any previous version, use

```
git diff commitID file
```

To revert one file to the previous version (aka cherry-pick the commit), use

```
git checkout commitID -- file
```

If you want to throw away some number of commits to start over, I strongly recommend against following any online directions recommending using `git reset --hard`. This command destructively removes revisions from your repository, and is responsible for many lost files that were not intended to be removed. The `revert` command is better and safer. It adds a new commit restoring the previous state, but your abandonded changes are still available in the previous commits if you need to cherry-pick individual file changes from those commits

```
git revert commitID
```

Submitting your changes

Your changes are not copied to the repository until you both commit them, **and** push them to the repository. You **must** do this for us to be able to check out and grade your work, but you can do it as many times as you want before then. Pushing your changes to the class repository before the deadline can help serve as a backup copy of your work, or allow you to work in multiple locations by copying changes to the main repository whenever you are done at one location before you move to the other. To push to the main repository, just do this:

```
git pull
git push
```

So we know exactly which commit you want us to grade, you'll be told a tag name to tag the commit for each assignment. To tag the latest commit with tag `assn1`, you'd use

```
git tag assn1
git push origin assn1
```

To tag a different commit (with commit ID found using git log), use

```
git tag assn1 commitID
git push origin assn1
```

If you realize you tagged the wrong commit, you need to remove the tag (locally, and on the server) before you retag. To remove a tag named `assn1`, you'd use this to remove the old tag before using one of the above methods to re-tag the submission.

```
git tag -d assn1
git push origin :refs/tags/assn1
```

Learning more

List of all Git commands

```
git
```

Help on any one Git command

```
git help command
```